Certification Support for Automatically Generated Programs

PROBLEM

Auto-generated code has many advantages
e.g., less coding effort, more maintainable
But: this technology comes with risk
• Can you trust the code-generator?
• How can the correctness of the generated code be assessed?
• Code generators are complex pieces of software themselves that may contain bugs

SOLUTION

Product-oriented certification

Don't try to verify the code-generator!

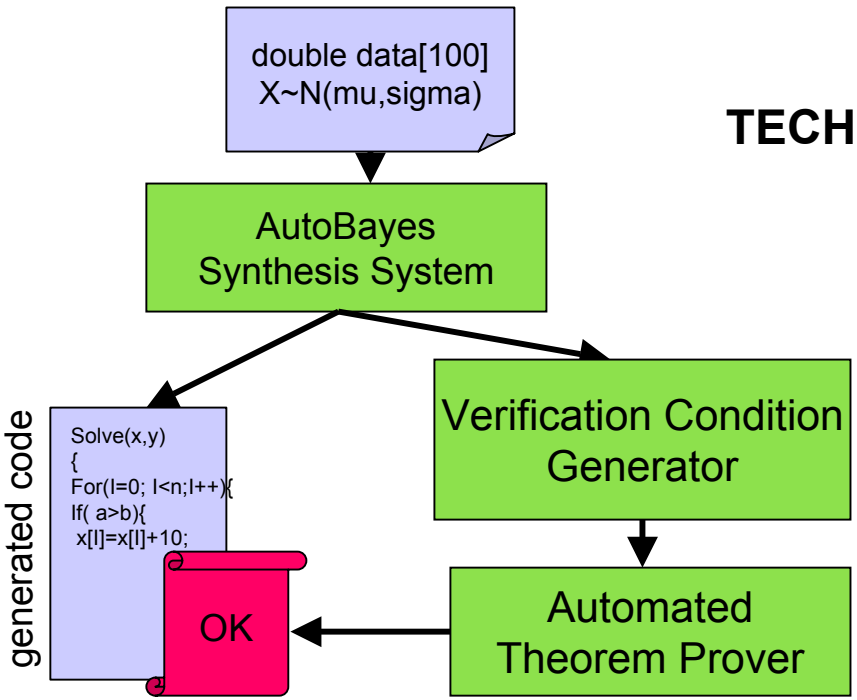
Rather: automatically verify safety properties for each generated piece of code, e.g.,

•Array-bound safety,

•Variable initialization safety,

• Division-by-zero, …

TECHNOLOGY



We extended the AutoBayes program synthesis system to *automatically* generate annotations for the selected properties. With a Verification Condition generator, we then produce logical conditions which are then automatically proven using an automated theorem prover. Thus, no user interaction is necessary. If required, each of the proofs could be checked (by an independent third party) for correctness.

Explanation of Accomplishment

- **POC:** Johann Schumann, (RIACS, ASE, Code IC, schumann@email.arc.nasa.gov)
- **Technology:** Although autocoding techniques promise large gains in software development productivity, their ``real-world'' application has been limited, particularly in safety-critical domains. Often, the major impediment is the missing trustworthiness of these systems: demonstrating---let alone formally certifying---the trustworthiness of automatic code generators is extremely difficult due to their complexity and size. We develop an alternative product-oriented certification approach which is based on five principles: (1) trustworthiness of the generator is reduced to the safety of each individual generated program; (2) program safety is defined as adherence to an explicitly formulated safety policy; (3) the safety policy is formalized by a collection of logical program properties; (4) Hoare-style program verification is used to show that each generated program satisfies the required properties; (5) the code generator itself is extended to automatically produce the code annotations required for verification. The approach is feasible because the code generator has full knowledge about the program under construction and about the properties to be verified. It can thus generate all auxiliary code annotations that an automated theorem prover needs to discharge all emerging verification obligations fully automatically.
- **Accomplishment:** This approach was used in a prototype certification extension for AutoBayes, an automatic program synthesis system which generates data analysis programs (e.g., for clustering and time-series analysis) from declarative specifications. In particular, we showed how a variable-initialization-before-use safety policy could be encoded and certified. The paper about this work “Certification Support for Automatically Generated Programs" (Johann Schumann, Bernd Fischer, Mike Whalen (U. Minnesota), and Jon Whittle) was presented by Johann Schumann at the 36th Hawaii International Conference on System Sciences (HICSS-36, 1/6-1/9, 2003) in the Software Engineering/Testing and Certification of Trustworthy Systems track. The paper was nominated for best paper in the Software Technology Track. The paper is available at http://ase.arc.nasa.gov/schumann/publications/papers/2003/hicss-36.html.
- **Benefits:** This technology has the potential to increase confidence in the use of code generators within and outside NASA. Auto-generated code will come with a certificate of its correctness (with respect to certain key properties that have been proved). These certificates can be independently checked by third parties such as a certification authority.